Revisiting Simple Neural Probabilistic Language Models

Simeng Sun and **Mohit Iyyer**
College of Information and Computer Sciences
University of Massachusetts Amherst
{simengsun, miyyer}@cs.umass.edu

Abstract

Recent progress in language modeling has been driven not only by advances in neural architectures, but also through hardware and optimization improvements. In this paper, we revisit the neural probabilistic language model (NPLM) of Bengio et al. (2003), which simply concatenates word embeddings within a fixed window and passes the result through a feed-forward network to predict the next word. When scaled up to modern hardware, this model (despite its many limitations) performs much better than expected on word-level language model benchmarks. Our analysis reveals that the NPLM achieves lower perplexity than a baseline Transformer with short input contexts but struggles to handle long-term dependencies. Inspired by this result, we modify the Transformer by replacing its first self-attention layer with the NPLM's local concatenation layer, which results in small but consistent perplexity decreases across three word-level language modeling datasets.

1 Introduction

Over the past decade, state-of-the-art neural architectures for language modeling (LM) have transitioned from simple recurrent neural networks (Mikolov et al., 2011) to LSTMs (Zaremba et al., 2014) and finally to Transformers (Vaswani et al., 2017). This progress is not due solely to LM-specific advances, however, as general-purpose upgrades such as residual connections (He et al., 2016) and layer normalization (Ba et al., 2016) have enabled scaling to huge datasets and model sizes (Kaplan et al., 2020) on powerful GPUs.

In this paper, we revisit the neural probabilistic language model (NPLM) of Bengio et al. (2003), the first (and simplest) neural architecture proposed for language modeling, through the lens of modern architecture design, hardware, and optimization. Given an input sequence of tokens, the NPLM first concatenates the previous n token embeddings and then passes the result through a feed-forward network to predict the next token. Due to its small context window and lack of parameter sharing, the NPLM has been rendered obsolete, discarded in favor of LSTMs and Transformers.

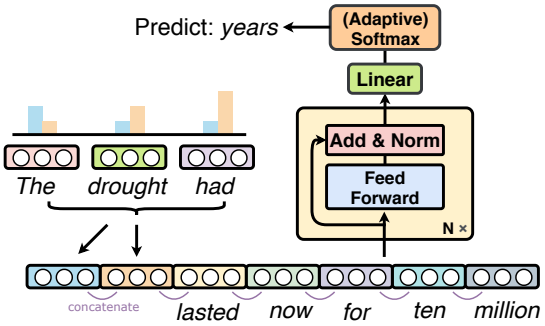

Figure 1: A modernized version of the neural probabilistic language model of Bengio et al. (2003), which concatenates token embeddings within a fixed local window and feeds them to a stack of feed-forward layers to predict the next token. Our modified version additionally concatenates representations of the distant context, which are computed by applying a weighted average to token representations outside the local window.

To what extent are its limitations mitigated by modern design and optimization choices? To answer this question, we design an upgraded NPLM featuring increased depth and window size n that incorporates residual connections, layer normalization, and dropout. We also include global context representations to the concatenation layer by applying simple aggregation functions to embeddings outside of the local context window. These modifications substantially improve the NPLM: on the WIKITEXT-103 benchmark dataset, the original NPLM of Bengio et al. (2003) reaches a validation perplexity of **216**, compared to **31.7** for our implementation, and **25.0** for a Transformer baseline.

Can we improve Transformer language models by hybridizing them with NPLMs? Interestingly, we discover that our NPLM actually *outperforms* the Transformer when given shorter input contexts

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5181–5188
June 6–11, 2021.

(Figure 2), although it is unable to take full advantage of longer contexts. Inspired by this result, we create two simple variants of the Transformer, one in which the first self-attention layer is replaced with the NPLM's concatenation layer, and the other in which self-attention in the first layer is constrained to a small local window.[1] These adjustments result in small but consistent perplexity decreases compared to a baseline Transformer across three word-level language modeling datasets (the first variant obtains **24.1** validation perplexity on WIKITEXT-103). Our qualitative analysis shows that the modified Transformers are better at predicting rare tokens and named entities, especially those that have already appeared in the context.

2 Neural probabilistic language models

Modern neural language models (NLMs) compute the conditional probability of a token w_t given preceding (or *prefix*) tokens $w_{<t}$ by first computing a dense vector representation of the prefix and then feeding it into a classifier to predict the next word. More concretely, a composition function g is applied to the sequence of token embeddings $\boldsymbol{x}_{<t}$ associated with the prefix, which results in a dense vector $\boldsymbol{z} = g(\boldsymbol{x}_{<t})$. A softmax classifier then takes $\boldsymbol{z}$ as input and produces a distribution $P(w_t \mid w_{<t})$ over the vocabulary. Transformers (Vaswani et al., 2017) are currently the most popular choice for the composition function g.

NPLM definition: First introduced by Bengio et al. (2003), the NPLM uses a simple composition function reminiscent of n-gram language modeling. It concatenates the last k prefix embeddings and passes the result through a feed-forward layer:

$$\boldsymbol{z} = \tanh(\mathbf{W}[\boldsymbol{x}_{t-k-1}; \boldsymbol{x}_{t-k} \ldots; \boldsymbol{x}_{t-1}]) \quad (1)$$

The NPLM has many intuitive limitations: (1) it ignores the global context provided by prefix tokens further than k tokens away; (2) it uses a different set of parameters for each position in the prefix window; and (3) it has a relatively small number of parameters, which limits its expressivity.

2.1 A modern update to the NPLM

To what extent are these limitations mitigated after scaling up the NPLM using modern advances in neural network training? Here, we investigate the impact of a number of modifications to the NPLM on WIKITEXT-103 validation perplexity (all results in Table 1).

Model	# Params	Val. perplexity
Transformer	148M	25.0
NPLM-old	32M[2]	216.0
NPLM-old (large)	221M[3]	128.2
NPLM 1L	123M	52.8
NPLM 4L	128M	38.3
NPLM 16L	148M	**31.7**
- Residual connections	148M	660.0
- Adam, + SGD	148M	418.5
- Global embedding	146M	41.9
- Global kernel, + average	148M	37.7
- Layer normalization	148M	33.0

Table 1: NPLM model ablation on WIKITEXT-103.

Increased depth and dimensionality: We pass the concatenated representation into a multi-layer network instead of a single layer, and we also substantially increase the embedding and hidden layer dimensionality to 410 and 2100 respectively. WIKITEXT-103 validation perplexity drops from **216** for the original one-layer NPLM (32M parameters) to **41.9** for a 16-layer NPLM with 148M parameters (no global prefix embeddings).

Better optimization for deep networks: To improve gradient flow across the multi-layer network, we apply residual connections (He et al., 2016) and layer normalization (Ba et al., 2016) at each layer. We additionally apply dropout (Srivastava et al., 2014), use rectified linear units (ReLU) instead of the $\tanh$ non-linearity, and train our NPLM with the Adam optimizer (Kingma and Ba, 2015).[4] These modifications are *crucial* for training our 16-layer NPLM: without residual connections, we reach a perplexity of **660**, while using standard SGD instead of Adam yields a perplexity of **418.5**.

Increased window size: While hardware considerations limited the window size k of the original NPLM to just five tokens, modern GPUs allow us to quickly train models with much larger memory footprints. We train models up to $k = 50$ (Figure 2)

[1] Code available at https://github.com/SimengSun/revisit-nplm

[2] Similar to (Bengio et al., 2003) we set embedding dimension to 60 and hidden dimension to 100.

[3] We use the same embedding dimension and hidden dimension of our modern NPLM model. Weights are not tied.

[4] Similar to Baevski and Auli (2019), we first linearly warm up learning rate for 4K steps and then anneal with one cycle cosine learning rate scheduler. We did not observe improvements annealing with cyclical scheduler.

and observe perplexity drop from **87** with $k = 3$ to eventually plateau around **40** with $k = 50$. The plot also shows that Transformers take far better advantage of longer inputs.

Tied weights and adaptive softmax: The original NPLM computes probabilities of *all* words in the vocabulary. For datasets with a large vocabulary, we use adaptive softmax (Grave et al., 2017) to speed up training and decrease the memory footprint. We also tie token embeddings with weights in the softmax layer (Press and Wolf, 2017) to further reduce model size. Without these modifications, our 16-layer NPLM does not fit in GPU memory, precluding training.[5]

Global context representation: Prior research demonstrates the effectiveness of representing large chunks of text using *averaged* token embeddings (Iyyer et al., 2015; Wieting et al., 2016). We leverage this work by applying a simple learned kernel (i.e., a 1-D convolution) to the prefix embeddings (beyond just the previous k) and including the resulting vector as an extra embedding to the concatenation layer. We also experiment with replacing the learned kernel with a uniform average. Adding these simple global embeddings improves the NPLM considerably: our 16-layer model's perplexity drops from **41.9** to **31.7** with the kernel-derived embedding, while the uniform average achieves a perplexity of **37.7**.

3 Using NPLMs to improve Transformers

While our upgraded NPLM achieves a massive perplexity reduction compared to the original implementation, it is still $\sim$ 6 perplexity points short of the baseline Transformer LM. Are there any takeaways from our results that can be used to *improve* Transformer LMs? In this section, we begin with an analysis experiment on WIKITEXT-103 that shows NPLMs outperform Transformers when given shorter prefixes. Inspired by this result, we propose two variants of a Transformer LM that integrate elements of the NPLM, and discover that both of them decrease perplexity across three word-level language modeling datasets (Table 2).

3.1 NPLMs are better with short contexts

Since NPLMs only concatenate a small, fixed number of prefix tokens together, they are obviously unsuited to handle global context. While our upgraded variant addresses this issue to some extent by including aggregated global prefix embeddings into the concatenation layer, the perplexity gap between NPLMs and Transformer LMs remains large. Here, we attempt to understand how much of this difference can be attributed to the Transformer's ability to better model global context. In particular, we train different NPLM and Transformer LMs by truncating the input prefix length to between 3 and 50 tokens. Our NPLM models do not have any global context embeddings in these experiments, and both the NPLM and Transformer models are 16 layers with $\sim$148M parameters each.

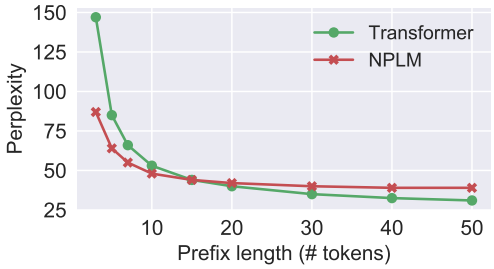

Figure 2: On the WIKITEXT-103 validation set, NPLM is better than the Transformer with short prefixes but worse on longer ones.

Figure 2 shows that NPLMs are actually *better* than Transformers when the input sequences are short (i.e., fewer than twenty prefix tokens), but as the prefixes get longer, NPLM perplexity plateaus, while the Transformer perplexity continually decreases. The plot shows that while multi-headed self-attention is effective for longer sequences, it may not be best for modeling shorter contexts.

3.2 Transformer variants

Inspired by these results, we investigate hybrid NPLM and Transformer models to better model both short and long-range contexts. In particular, we create two variants of the Transformer by modifying only its *first* layer (L0), while keeping every other layer the same. In the first modification, **Transformer-N**, we simply replace the first self-attention block in L0 with the NPLM's local concatenation layer (Equation 1), without including any global embeddings. Wondering if the behavior of the concatenation layer can be replicated by self-attention, we also design **Transformer-C**, in which the self-attention window in L0 is constrained to the previous 5 tokens. This constraint is similar to the windowed attention approaches pre-

[5]Our models are trained on 4 GeForce GTX 1080Ti GPUs.

	WIKITEXT-2 (13M)		WIKITEXT-103 (148M)		LAMBADA (115M)		ENWIK8 (38M)	
	Valid ppl.	Test ppl.	Valid ppl.	Test ppl.	Valid ppl.	Test ppl.	Valid bpc.	Test bpc.
NPLM	120.5	114.3	31.7	32.9	44.8	44.5	1.63	1.63
Transformer	117.6	111.1	25.0	26.1	42.1	41.8	**1.14**	**1.12**
Transformer-C	113.1	107.5	24.1	**25.1**	42.0	41.7	1.14	1.12
Transformer-N	**110.8**	**105.6**	**24.1**	25.2	**41.8**	**41.5**	1.14	1.12

Table 2: Our Transformer variants improve on the baseline Transformer across three word-level LM datasets. The # of model parameters is shown in brackets (same for all models). For model details, see Appendix B.

viously applied at all layers in prior Transformer variants (Beltagy et al., 2020; Roy et al., 2020).[6]

3.3 Experimental details

Datasets We evaluate our models on four language modeling datasets: WIKITEXT-2 and WIKITEXT-103 (Merity et al., 2016), LAMBADA (Paperno et al., 2016), and the character-level ENWIK8 benchmark (Merity et al., 2017). For WIKITEXT-2 and WIKITEXT-103 (Merity et al., 2016), we insert an <eos> token after each line, following Merity et al. (2018). We use adaptive softmax (Grave et al., 2017) on WIKITEXT-103 with cutoffs $(2e4, 4e4, 2e5)$. On LAMBADA, we follow Paperno et al. (2016) by considering only the most frequent 60K words and replacing the rest with <unk> tokens. We use the preprocessing script released by Merity et al. (2017) to process ENWIK8.

Models We train 16-layer (16L) models on the larger WIKITEXT-103 and LAMBADA datasets, 12L models for ENWIK8, and 6L for the small WIKITEXT-2 dataset.[7] For each dataset, we scale embedding and hidden dimensionality to ensure that all models have roughly the same number of parameters. After tuning hyperparameters on the validation data, we set the number of local concatenated tokens to 15 and the number of 1-D convolution kernels to 5.

Training details Our NPLM is trained with dropout probability $p = 0.2$, while the other models use $p = 0.1$ on all datasets except for WIKITEXT-2, for which they use $p = 0.3$. For all models, we use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and training is conducted on 1080Ti GPUs. During evaluation, we follow the methodology of (Khandelwal et al., 2020) by providing extra prior context for the scored tokens, for instance, in a block of 512 tokens, only the last 128 tokens are scored with the first 384 tokens as context. Detailed architecture, training, and evaluation configurations are included in Appendix B.

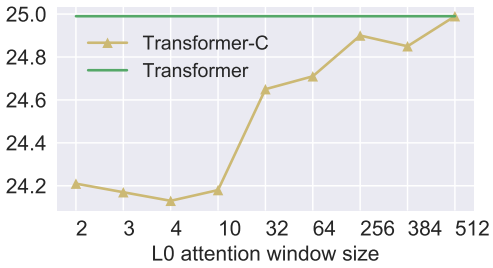

Figure 3: Transformer-C perplexity decreases with small L0 attention windows.

3.4 Results and analysis

Table 2 shows that **Transformer-N** improves over the baseline Transformer across all three word-level language modeling benchmarks, with the biggest perplexity drop coming on the small WIKITEXT-2 dataset, although character-level perplexity on ENWIK8 is unchanged. **Transformer-C** also outperforms the baseline Transformer but by smaller margins than **Transformer-N**.

Narrower window size in L0 is better: We examine WIKITEXT-103 val. perplexity as a function of Transformer-C window size. Figure 3 shows drops of ~ 1 perplexity point with window sizes of 2-4, which disappear as window size is increased. This experiment supports the importance of focusing on local context at lower layers.

Hybrid models improve at predicting entities and rare words: To obtain a more fine-grained understanding of our models, we turn to the long-distance dependency prediction task in LAMBADA (Paperno et al., 2016), a manually-annotated subset of the full dataset in which correctly predict-

[6] We do not observe improvements when using local attention at all layers.

[7] The relatively high WIKITEXT-2 perplexities are likely because we did not apply separate regularization that Merity et al. (2017) show is useful for such a small dataset.

Model	Test	Control	CF	LF	Ent.
NPLM	0.40	30.46	-	-	-
Transformer	30.60	35.84	38.94	29.47	32.26
Transformer-N	**32.51**	37.06	42.33	30.14	33.95
Transformer-C	32.23	**37.34**	**42.65**	**31.58**	**35.03**

Table 3: NPLM and Transformer variants on LAMBADA target word accuracy (%). Variants perform better on context-frequent (CF) tokens that appear at least twice in previous context, low frequency (LF) tokens with frequency < 1500, and named entities (Ent).

ing a token is possible only when longer contexts are provided.

Table 3 shows that our upgraded NPLM achieves less than 1% accuracy (argmax prediction) on the test set but 30% on a control set that does not test long-term dependencies. As the baseline Transformer reaches over 30% accuracy on the test set, this result shows that the convolutional kernels in our modernized NPLM are incompetent at modeling long-range context.

On the other hand, both **Transformer-N** and **Transformer-C** outperform the baseline Transformer (Table 3) by over 1.5% on the test set. To better understand these improvements, we perform a fine-grained analysis of the tokens for which these models improve over the Transformer. This analysis reveals that the gains stem mainly from three types of target tokens: (1) context-freqeunt (CF) tokens that appear more than twice in the prefix; (2) low frequency tokens (LF) with frequency below 1500; and (3) named entity tokens (Ent) detected by the spaCy (Honnibal et al., 2020) NER tagger. The three right-most columns of Table 3 shows that both Transformer variants are more accurate at predicting these tokens, which demonstrates the benefits of enforcing local focus at the first layer.

4 Related work

The NPLM model in this paper based entirely on the original formulation from Bengio et al. (2003). The variants in our analysis are based on the Transformer model (Vaswani et al., 2017) and Transformer LMs (Baevski and Auli, 2019; Dehghani et al., 2019; Dai et al., 2019; Sukhbaatar et al., 2019; Khandelwal et al., 2020; Wang et al., 2019; Press et al., 2020a; Mandava et al., 2020; Press et al., 2020b). The constrained local attention in Transformer-C is adopted at all layers of models such as Longformer (Beltagy et al., 2020) and Big Bird (Zaheer et al., 2020) due to its sparsity. Our work conceptually resembles that of Chiu and Rush (2020), who modernize HMM language models, as well as simple RNN-based language models (Merity et al., 2018). Our linguistic analysis is inspired by experiments from Khandelwal et al. (2018).

5 Conclusion

We discover that general-purpose advances in neural architecture design, hardware, and optimization significantly improve the NPLM, a classic language model. An analysis of our upgraded NPLM inspires us to hybridize it with a modern Transformer LM and obtain perplexity decreases across three word-level LM datasets.

Ethics statement

Misuse of language models Our research involves training large language models on publicly available benchmark datasets. They share the same issues faced by many pretrained language models, such as being used maliciously to generate unfaithful, biased or offensive output.

Energy costs We train our models and variants on 4 GeForce GTX 1080 Ti GPUs for all datasets except WIKITEXT-2. We use only one GPU for experiments on WIKITEXT-2. The Transformer and its variants take longer to train (40h, 102h, and 108h on WIKITEXT-103, LAMBADA, and ENWIK8 respectively). Our modernized NPLM does not have attention module, and therefore trains relatively faster (32h, 45h, and 88h for the above datasets). The energy costs of training and tuning these models, as well as doing exploratory experiments in the initial stages of the project, cannot be ignored. That said, compared to Transformer models, the modernized NPLM has significantly reduced training time, and hence carbon costs. We hope our work contains useful insights for future research that aims to develop simpler and more efficient language models.

Acknowledgements

We thank Nader Akoury, Andrew Drozdov, Shufan Wang, and the rest of UMass NLP group for their constructive suggestions on the draft of this paper. We also thank the anonymous reviewers for their helpful comments. This work was supported by award IIS-1955567 from the National Science Foundation (NSF).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Yoshua Bengio, R. Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Justin Chiu and Alexander Rush. 2020. Scaling hidden Markov language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1341–1349, Online. Association for Computational Linguistics.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *International Conference on Learning Representations*.

Édouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2017. Efficient softmax approximation for GPUs. volume 70 of *Proceedings of Machine Learning Research*, pages 1302–1310, International Convention Centre, Sydney, Australia. PMLR.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Swetha Mandava, Szymon Migacz, and Alex Fit Florea. 2020. Pay attention when required. *arXiv preprint arXiv:2009.04534*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and optimizing LSTM language models. *CoRR*, abs/1708.02182.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An analysis of neural language modeling at multiple scales. *CoRR*, abs/1803.08240.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5528–5531. IEEE.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Ofir Press, Noah A. Smith, and Omer Levy. 2020a. Improving transformer models by reordering their sublayers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2996–3005, Online. Association for Computational Linguistics.

Ofir Press, Noah A. Smith, and Mike Lewis. 2020b. Shortformer: Better language modeling using shorter inputs.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Aurko Roy, M. Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient content-based sparse attention with routing transformers. *ArXiv*, abs/2003.05997.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 331–335, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Chenguang Wang, Mu Li, and Alexander J. Smola. 2019. Language models with transformers. *CoRR*, abs/1904.09408.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization.

A Experiment details

Dataset	Train #Tokens	Vocab. size
WIKITEXT-2	2M	33K
WIKITEXT-103	103M	267K
LAMBADA	203M	60K
ENWIK8	100M	205

Table 4: Dataset statistics

Dataset statistics are shown in Table 4.

Dataset	Train len	Test len	Tgt. len
WIKITEXT-2	512	512	128
WIKITEXT-103	512	512	128
LAMBADA	512	512	128
ENWIK8	1024	1024	512

Table 5: Training sequence length as well as scored target length and total test sequence length during evaluation we used on each dataset.

Evaluation We follow the practice in (Khandelwal et al., 2020) to provide extra prior context for the scored tokens. We provide the training sequence length, test total sequence length, and test target sequence length in Table 5.

B Model configurations

Detailed model configurations are shown in Table 6. Training details are shown in Table 7.

	WIKITEXT-2		WIKITEXT-103		ENWIK8		LAMBADA	
	NPLM	Transformer	NPLM	Transformer	NPLM	Transformer	NPLM	Transformer
# Layers	6	6	16	16	12	12	16	16
Emb. dimension	256	256	410	410	512	512	512	512
Hidden dimension	1024	1024	2100	2100	2048	2048	4096	4096
Concat hidden dimension	400	-	2000	-	1400	-	2000	-
# Attention heads	-	4	-	10	-	8	-	16
Adaptive softmax	no	no	yes	yes	no	no	no	no
# Concat tokens	15	-	15	-	15	-	15	-
# Kernel global	5	-	5	-	5	-	5	-
Dropout	0.3	0.3	0.2	0.1	0.2	0.1	0.2	0.1
#Param	13M	13M	149M	148M	38M	38M	115M	115M

Table 6: Model configuration on WIKITEXT-2 , WIKITEXT-103 , ENWIK8 , LAMBADA .

	Warmup steps	Learning rate	Max steps	Batch size	Training time
WIKITEXT-2	100	5e-4	10k	5120	1.2h/1h
WIKITEXT-103	4k	2.5e-4/3.5e-4	200k	10240	40h/32h
ENWIK8	0	2.5e-4	400k	22528	102h/45h
LAMBADA	4k	3e-4	400k	8192	108h/88h

Table 7: Details of training on the four datasets. Models are trained on single 1080Ti GPU for WIKITEXT-2, and on four 1080Ti GPUs for the rest datasets. When a configuration is different for Transformer and NPLM, it's shown in the order Transformer/NPLM.